**基本Docker的分布式应用控制系统**

**关键代码**

**一、Restful PHP**

将restful请求方式应用于PHP，通过get/post/delete等请求方法像Docker Server发送请求。

protected function request\_get($cmd, $params = [])

{

return self::curl($this->Host . $cmd, $params);

}

protected function request\_post($cmd, $params = [], $gets = [])

{

return self::curl($this->Host . $cmd, $gets, $params, true);

}

protected function request\_delete($cmd)

{

$service\_url = $this->Host . $cmd;//'http://192.168.109.128:4232/images/alpine:latest';

$ch = curl\_init($service\_url);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, "DELETE");

curl\_setopt($ch, CURLOPT\_POSTFIELDS);

$response = curl\_exec($ch);

curl\_close($ch);

return $response;

}

protected function curl($url, $gets = [], $posts = [], $POST = false)

{

$ch = curl\_init();

$url = str\_replace('//', '/', $url);

$URL = $gets ? $url . '?' . http\_build\_query($gets) : $url;

$optBool = curl\_setopt\_array($ch, [

CURLOPT\_URL => $URL,

CURLOPT\_RETURNTRANSFER => true

]);

if ($posts) {

$POST = true;

$POST\_DATA = is\_string($posts) ? $posts : json\_encode($posts, JSON\_UNESCAPED\_UNICODE);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $POST\_DATA);

}

if ($POST) {

curl\_setopt($ch, CURLOPT\_POST, 1);

curl\_setopt($ch, CURLOPT\_HTTPHEADER, [

'Content-Type: application/json',

]);

}

$rs = curl\_exec($ch); //重点语句

$info = curl\_getinfo($ch);

curl\_close($ch);

$imageData = [];

if (preg\_match('/Downloaded newer image/', $rs)) {

$imageData['status'] = true;

$imageData['info'] = 'Image created successfully!';

} else if (preg\_match('/Already exists/', $rs)) {

$imageData['status'] = false;

$imageData['info'] = 'Image already exists!';

} else if (preg\_match('/Network timed out/', $rs)) {

$imageData['status'] = false;

$imageData['info'] = 'Network timed out!';

} else {

$imageData = $rs && substr($rs, 0, 1) == "{" ? json\_decode($rs, true) : $rs;

}

return $imageData;

}

1. **Remote API**

Remote API 使用对照表

|  |  |  |
| --- | --- | --- |
| **操作** | **Remote API** | **请求方式** |
| List containers | /containers/json | GET |
| Create a container | /containers/create | POST |
| Inspect a container running inside a container | /containers/(id)/json | GET |
| List processes | /containers/(id)/top | GET |
| Get container logs | /containers/(id)/logs | GET |
| Inspect changes on a container’s filesystem | /containers/(id)/changes | GET |

Remote API通过 Restful PHP请求方法使用，关键代码如下：

/\*\*

\* 获取容器列表信息

\* @return mixed

\*/

function getContainers()

{

return json\_decode(self::request\_get('/containers/json'), true);

}

/\*\*

\* inspect a container

\* @param string $ID 容器ID

\* @return mixed

\*/

function inspectContainer($ID)

{

$rs = self::request\_get("/containers/{$ID}/json");

return $rs;

}

/\*\*

\* 根据容器ID获取容器进程列表

\* @param $ID 容器ID

\* @return mixed

\*/

function getContainerProcess($ID)

{

return self::request\_get("/containers/{$ID}/top");

}

/\*\*

\* start a container

\* @param $ID containerId

\* @return bool

\*/

function startContainer($ID)

{

$rs = self::request\_post("/containers/{$ID}/start");

return $rs == "" ? ['success' => true] : ['success' => false, 'error' => $rs];

}

1. **控制器应用**

控制器应用主要是根据前端发来的数据，向Docker Server发送请求，关键代码如下：

/\*\*

\* 获取容器信息列表，多条件查询

\* param containerStatus String 容器状态，查询条件之一，包括：created/running/stopped/restarting/paused/exited/dead

\* param containerName String 容器名称，查询条件之一

\* param hostName String 主机名称，查询条件之一

\* param pageSize int 分页大小， 默认为 20

\* param nowPage int 当前页，默认为 1

\* return success Bool 请求是否成功 true/false

\* return containerList int 容器列表 请求成功将返回该字段

\* return error String 失败信息 请求失败将返回该字段

\*/

public function searchContainerList() {

$containerStatus = I('get.containerStatus');

$containerName = I('get.containerName');

$hostName = I('get.hostName');

if($containerStatus != "") {

$searchData['container\_status'] = $containerStatus;

}

if($containerName != "") {

$searchData['container\_name'] = $containerName;

}

if($hostName != "") {

$searchData['host\_name'] = $hostName;

}

$pageSize = I('get.pageSize') == "" ? 20 : I('get.pageSize');

$nowPage = I('get.nowPage') == "" ? 1 : I('get.nowPage');

$ContainerInfoModel = new ContainerListViewModel();

$containerList = $ContainerInfoModel->where($searchData)->page($nowPage, $pageSize)->select();

$count = $ContainerInfoModel->where($searchData)->count();// 查询满足要求的总记录数

$returnData = [

'containerList' => $containerList,

'count' => $count

];

$this->ajaxReturn($returnData);

}

/\*\*

\* 创建容器

\* param imageName Must 镜像名称

\* param hostAddress Must 主机地址

\* param hostId Must 主机编号

\* @return bool

\*/

public function createContainer() {

$returnData = [];

$imageName = $\_POST['imageName'];

$hostAddress = $\_POST['hostAddress'];

$hostId = $\_POST['hostId'];

if($imageName == "" || $hostAddress == "" || $hostId == "") {

$returnData['success'] = false;

$returnData['error'] = "参数不足！请保证镜像名称、主机地址、主机编号均存在！";

$this->ajaxReturn($returnData);

}

//创建容器

$DockerSDK = new DockerSDKService($hostAddress);

$containerInfo = $DockerSDK->createContainer(

$imageName,[

'22/tcp'=> $DockerSDK->object

],

'date',[]

);

if($containerInfo === false) {

$returnData['success'] = false;

$returnData['error'] = "主机地址错误！";

$this->ajaxReturn($returnData);

}

//向Docker服务发送get请求container信息，并将信息存入数据库

$data = $DockerSDK->inspectContainer($containerInfo['Id']);

$ContainerInfoModel = new ContainerInfoModel();

$saveData['container\_id'] = $data['Id'];

$saveData['container\_name'] = substr($data['Name'], 1);//去掉斜杠

$saveData['container\_status'] = $data['State']['Status'];

$saveData['image'] = $imageName;

$saveData['command'] = implode(";", $data['Config']['Cmd']);

$saveData['ports'] = $data['NetworkSettings']['Ports'];

$saveData['created\_time'] = $data['Created'];

$saveData['logs'] = '';

$saveData['others'] = json\_encode($data, true);

$saveData['host\_id'] = $hostId;

$addBool = $ContainerInfoModel->data($saveData)->add();

if(!$addBool) {

$returnData['success'] = false;

$returnData['error'] = "数据库保存容器失败！";

}else {

$returnData['success'] = true;

}

$this->ajaxReturn($containerInfo);

}